Weakly-hard Real-time Guarantees for Earliest Deadline First Scheduling of Independent Tasks

Zain a H Hammadeh, Sophie Quinton, Rolf Ernst

Weakly-Hard Real-Time Guarantees for Earliest Deadline First Scheduling of Independent Tasks

ZAIN A. H. HAMMADEH, TU Braunschweig, Germany
SOPHIE QUINTON, INRIA Grenoble - Rhône-Alpes, France
ROLF ERNST, TU Braunschweig, Germany

The current trend in modeling and analyzing real-time systems is toward tighter yet safe timing constraints. Many practical real-time systems can de facto sustain a bounded number of deadline-misses, i.e., they have Weakly-Hard Real-Time (WHRT) constraints rather than hard real-time constraints. Therefore, we strive to provide tight Deadline Miss Models (DMMs) in complement to tight response time bounds for such systems. In this work, we bound the distribution of deadline-misses for task sets running on uniprocessors using the Earliest Deadline First (EDF) scheduling policy. We assume tasks miss their deadlines due to transient overload resulting from sporadic jobs, e.g., interrupt service routines. We use Typical Worst-Case Analysis (TWCA) to tackle the problem in this context. Also, we address the sources of pessimism in computing DMMs, and we discuss the limitations of the proposed analysis. This work is motivated by and validated on a realistic case study inspired by industrial practice (satellite on-board software) and on a set of synthetic test cases. The synthetic experiment is dedicated to extensively study the impact of EDF on DMMs by presenting a comparison between DMMs computed under EDF and Rate Monotonic (RM). The results show the usefulness of this approach for temporarily overloaded systems when EDF scheduling is considered. They also show that EDF is especially well for WHRT tasks.

ACM Reference Format:
Zain A. H. Hammadeh, Sophie Quinton, and Rolf Ernst. 2019. Weakly-Hard Real-Time Guarantees for Earliest Deadline First Scheduling of Independent Tasks. 1, 1 (June 2019), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In real-time systems, temporal constraints often expressed as deadlines have to be satisfied to guarantee the correction of system results. When multiple tasks share one resource in a real-time system, scheduling is used to resolve the contention between tasks: the operating system follows a *scheduling policy* to arbitrate between requests to access the shared resource. If there is a scheduling policy such that all tasks in a task set meet all deadlines, there is a *feasible schedule* for the task set.

Earliest Deadline First (EDF) [24] is a scheduling policy which gives the highest priority to those tasks that are the closest to missing their deadline. EDF scheduling has been proved to be *optimal* [13] in the sense of feasibility under certain conditions. If there exists a feasible schedule for a task set, then EDF scheduling can find a feasible schedule as well. Verifying that a given schedule is feasible (all tasks in a task set meet all their deadlines), is called *schedulability analysis*.

Authors' addresses: Zain A. H. Hammadeh, TU Braunschweig, Germany, hammadeh@ida.ing.tu-bs.de; Sophie Quinton, INRIA Grenoble - Rhône-Alpes, France, sophie.quinton@inria.fr; Rolf Ernst, TU Braunschweig, Germany, ernst@ida.ing.tu-bs.de.

XXXX-XXXX/2019/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Safety-critical real-time systems require *hard* real-time guarantees. That is, schedulability analysis must guarantee that all deadlines are met, as otherwise the functionality of the system may be jeopardized. However, many practical embedded real-time systems can de facto sustain a bounded number of deadline-misses. In control engineering, which is the natural application domain of real-time computing, papers have been published [10, 16, 32, 41] showing that it is sometimes possible to guarantee the performance of a control system despite some missed deadlines. Such systems are called *weakly-hard* real-time systems [4]. Formally, a WHRT system is a system in which the sequence of met and missed deadlines of any consecutive k deadlines is precisely bounded. Therefore, the guarantees provided by a weakly-hard schedulability analysis must satisfy the constraints on such sequences. The notation (m, k) – introduced in [19] as (m, k)-firm – is used to define a constraint which bounds the allowed number of deadline-misses for an individual task such that there are no more than m admissible deadline-misses out of any consecutive k deadlines.

There are three main sources of pessimism in a schedulability analysis: (1) tasks do not always run for their worst-case execution time (wcet) due to not following the worst-case path in the task control flow graph, and due to the pessimism of the wcet analysis itself, (2) tasks are activated less often than in the worst-case activation model assumed for schedulability analysis and (3) tasks do not suffer every time the worst-case blocking. As a result, for many systems, almost all deadlines are actually met at run-time, even though the analysis cannot exclude some deadlines misses [3]. State-of-the-art regarding WHRT [4, 20, 44] includes several analyses that provide (m, k) guarantees considering different system models. In this work, we aim to provide (m, k) guarantees for WHRT systems when EDF scheduling is considered.

In real-world applications, *overload conditions* – critical situations in which some deadlines may be missed – may occur even when the system is properly designed: reasons for this can be changes in the environment, simultaneous arrivals of asynchronous events, faults of peripheral devices, or system exceptions [8]. Under overload conditions, experiments in [30] have shown that tasks under EDF scheduling can miss many deadlines because EDF scheduling assigns the highest priority to the tasks which are the closest to missing their deadlines [8]. In the extreme case, all tasks in the system may miss their deadline; this phenomenon is called *domino effect*. The domino effect can jeopardize the system functionality as even the most critical tasks miss their deadline. Note that, the analysis proposed in this paper applies to systems for which the overload is due to reasons that are known at design time: Neither reliability analysis, nor fault tolerant system design is in the scope of this paper. Reliability analysis [2] computes the likelihood of missing deadlines due to the effects of errors, while other solutions aim to design the system such that more errors can occur without compromising deadlines [22, 28].

In this paper, we are interested in computing (m, k) guarantees for WHRT tasks which may miss their deadlines due to transient overload resulting from sporadic jobs, e.g., interrupt service routines, when EDF scheduling is considered, using Typical Worst-Case Analysis (TWCA) [20]. In TWCA, the lifetime of a resource is partitioned into 1) typical intervals where there is no transient overload and 2) temporarily overloaded intervals due to the transient overload [37]. TWCA handles the two types of intervals separately and bounds the impact of the sporadic jobs in terms of deadline-misses within the temporarily overloaded intervals. TWCA considers arrival curves as activation models, arbitrary deadlines, and it is applicable so far to uniprocessor systems under fixed priority preemptive or non-preemptive schedulers. TWCA relies on existing worst-case response time analyses as a foundation. Therefore, for our extension to EDF, we need to call the worst-case response time analysis for EDF scheduling. EDF scheduling has been studied widely to find more efficient schedulability tests [17, 42, 46]. In this paper, we focus instead on worst-case response time analysis.

The major contribution in this work is an extension of TWCA as presented in, e.g. [20, 45] to EDF scheduling to bound the distribution of the system met and missed deadlines in the form of (m, k) guarantees. To the best of our knowledge, this is the first attempt to consider EDF in WHRT systems. We show in this paper that TWCA is extendible to EDF with no need to develop a solution from scratch. Experimental results show that TWCA is a useful approach to compute guarantees for WHRT systems when EDF scheduling is considered.

The rest of the paper is organized in 8 sections as follows: In Section 2, we present a perusal of the related work. Section 3 shows the system model, which is considered in this paper. In Section 4, we call the worst-case response time analysis for EDF scheduling, and we call state-of-the-art TWCA in Section 5. Section 6 contains the main contribution of the paper, namely the computation of DMMs. We address the sources of pessimism in computing DMMs in Section 7. In Section 8, we motivate and validate our analysis on a realistic case study (satellite on-board software), and on a set of synthetic test cases to extensively test our analysis. Finally, we conclude our work in Section 9.

2 RELATED WORK

WHRT systems have received quite a lot of attention in the last four years. The term *weakly-hard*, though, is quite a bit older than this: it was coined in [4], where a WHRT system is defined as a system in which tasks can tolerate at most m deadline-misses out of any consecutive k deadlines. The notation (m, k) to represent the tolerated bound is even older and originates from the work on (m, k)-firm systems [19], which addresses the same type of systems. The notation (m, k)-firm was invested in [36, 38]. Authors took advantage of tolerating deadline-misses to manage overload in control applications. Their system model considers periodic tasks with fixed priority preemptive scheduling.

In [4], weakly-hard guarantees are computed for periodic tasks with fixed priorities. [44] extended [4] to offset-free periodic tasks. That work is proper for systems where only small k matters and, however, it does not scale beyond 20 tasks and $k > 10$ because of the complexity of their proposed MILP [34]. In this work, we consider arrival curves to describe activation models. Also, the experiments show that TWCA scales beyond 20 tasks and $k > 1000$ for Fixed Priority Preemptive (FPP) and EDF.

Kumar et al. proposed an analysis in [25] in the context of real-time calculus [11]. The presented analysis computes the settling time, i.e., the longest time window after the rare event until the system returns to normal. In addition, the overshoot during the settling time quantifies how many deadline-misses may then occur. The main difference with TWCA is that [25] considers only one source of rare events at a time. In [7], some predefined tasks are allowed to miss their deadline occasionally in uncertain or faulty execution conditions due to soft errors. Dynamic real-time guarantees were computed to determine if the system can provide full timing guarantees or limited timing guarantees and to determine the maximum interval length until the system will again provide full timing guarantees. The system model considers sporadic independent tasks in a uniprocessor system under a fixed-priority scheduling policy.

State-of-the-art TWCA [20] provides WHRT guarantees on the distribution of deadline-misses in the form of (m, k) for uniprocessor system under fixed priority preemptive or non-preemptive. In [21], the system model was extended to bound deadline-misses in WHRT systems with task dependencies. In this work, we present an extension of TWCA [20, 45] to provide WHRT guarantees for independent tasks when EDF scheduling is considered.

The problem of providing bounds on deadline-misses was also addressed by the *probabilistic* real-time analyses where a random variable describes at least one parameter. In [14] and [31], stochastic analysis of periodic real-time systems was presented. The analysis applies to uniprocessor

system under FPP or EDF scheduling. The probability that a task misses its deadline in an infinite window size was computed. In [9], a *probabilistic deadline-miss analysis* is presented for periodic tasks sharing a uniprocessor under fixed priority non-preemptive scheduling policy. The analysis supports the derivation of the probability that a deadline-miss occurs within time t. In that work, jobs are discarded as soon as their deadline is missed. In [40], a *probabilistic calculus* is presented. Santinelli and Cucu-Grosjean developed in [40] an analysis in terms of sufficient probabilistic schedulability conditions for task systems with either FPP or EDF scheduling policies. [40] follows the compositional performance analysis. Therefore, it applies to distributed real-time systems. The probability of missing a deadline of a periodic or non-periodic task in an infinite window size can be derived from the proposed analysis. With the same system model used in [7] and based on probabilistic WCETs, [12] calculates the probability of m-consecutive deadline-misses of a task in faulty execution conditions due to soft errors. However, probabilistic deadline guarantees are not sufficient for real-time systems such as automatic control systems [35]. Instead, a precise bound on the distribution of the system met and missed deadlines during a time window is necessary and this can be done using (m, k)-firm model [27].

Establishing an on-line scheduling framework for WHRT systems has been addressed in [3]. Bernat and Cayssials presented an on-line scheduling framework called Bi-Modal Scheduler (BMS). It is characterized by two modes of operation. Weakly hard constraints are guaranteed to be satisfied by switching, whenever necessary, from a normal mode to a panic mode for which schedulability tests exist that guarantee the constraints on the allowed number of deadline-misses. Similarly, [27] proposed on-line (m, k)-firm enforcement policy for control systems with non-preemptive EDF scheduling. Establishing an on-line scheduling is out of the scope of this paper.

3 SYSTEM MODEL

We consider a *uniprocessor* system running a finite set of real-time tasks $\mathcal{Z}$, which compete for processor time to perform some computations. On a single processor, all durations share a common time unit, namely the clock cycle. As a consequence, all parameters in this paper have positive integer values, unless stated otherwise.

Tasks are activated by a timer or the occurrence of some event. Each activation of a task results in the creation of a *job*, which corresponds to some computation to be performed. Upon creation, a job requests access to the processor in order to execute. That access is granted by the scheduler according to its policy — in our case EDF, such that the job with the earliest deadline is *scheduled*, i.e., granted access to the processor. We consider *preemptive* EDF scheduling, so the execution of a job may be interrupted by the activation of another job with an earlier deadline.

The timing characteristics of a task are: (1) the instants at which the task is activated; (2) how much processor time it needs to complete each job; and (3) its relative deadline, i.e., how long after activation each job must be completed. As a result, tasks are defined as follows.

Definition 3.1. A *task* $\tau_i \in \mathcal{Z}$ is defined by:

(1) an activation model defined using *arrival curves*, see Definition 3.2;
(2) an upper bound on its execution time C_i;
(3) and a relative deadline D_i.

Note that we make no assumptions w.r.t deadlines, i.e., we consider so-called *arbitrary deadlines*.

Definition 3.2. Arrival curves are functions $\eta_i^+, \eta_i^- : \mathbb{N} \rightarrow \mathbb{N}$ that bound the number of activations of a task τ_i, such that $\eta_i^+(\Delta)$ (respectively $\eta_i^-(\Delta)$) upper (respectively lower) bounds the number of activations of τ_i that may occur within any time interval of length Δ. We sometimes use pseudo-inverse representations of arrival curves, namely $\delta_i^-, \delta_i^+ : \mathbb{N} \rightarrow \mathbb{N}$, such that $\delta_i^-(k)$ (respectively

$\delta_i^+(k)$) lower (respectively upper) bounds the length of any time interval containing k activations of τ_i.

Note that two versions of arrival curves have been defined [39], depending on whether the considered time intervals are right-open or closed. While the analysis for FPP uses right-open intervals, the analysis for EDF scheduling is based on closed intervals. In the rest of the paper, we use the notations introduced above for right-open intervals, and the notations $\tilde{\eta}_i^+(\Delta)$, $\tilde{\eta}_i^-(\Delta)$[1] whenever closed intervals are considered in bounding the number of jobs of τ_i.

A job ℓ of τ_i is activated at act_i^ℓ, and it completes execution at end_i^ℓ. For each job we define an *absolute deadline* d_i^ℓ such that: $d_i^\ell = act_i^\ell + D_i$. The *response time* R_i^ℓ of a job is the duration between its activation and its completion: $R_i^\ell = end_i^\ell - act_i^\ell$. A deadline-miss occurs if the response time of a job exceeds the relative deadline of its corresponding task. The worst-case response time R_i^+ of a task τ_i is then the maximum response time among all possible jobs of τ_i.

Definition 3.3 (Utilization). The computation of utilization of a task τ_i requires the evaluation a limit for time approaching infinity.

$$U_i = \lim_{\Delta t \to \infty} \frac{\eta_i^+(\Delta t) \,.\, C_i}{\Delta t} \tag{1}$$

the resource utilization is then:

$$U = \sum_{\tau_i \in \mathcal{Z}} U_i \tag{2}$$

This is a generalization of the standard definition of utilization for periodic tasks.

3.1 Problem formulation

We are interested in systems that are *temporarily overloaded*: Such systems may suffer transient overload situations and even miss deadlines, but their utilization U is smaller than or equal to 1. Systems that do not satisfy this condition consistently need more computation resources than they have.

When a transient overload causes a deadline-miss, it is said to be an *active fault*, and the deadline-miss is an *error*. The error may cause a *failure*, i.e., the system delivers an incorrect service [1]. If the error causes no failure, it is said to be *tolerable*.

Definition 3.4. A task τ_i tolerates — without error handling— at most m deadline-misses out of any k consecutive jobs, and no failure will be triggered unless τ_i misses $m+1$ deadlines out of any k consecutive jobs where $m \geqslant 0$ and $k \geqslant m$.

In other words, missing m deadlines out of any k consecutive jobs will never cause a failure.

The definition of hard real-time systems can be derived by saying it is a system in which $m = 0$ for all tasks. A real-time system whose tasks can tolerate few deadline-misses out of a sequence of jobs is known as a weakly-hard real-time system.

Definition 3.5 (WHRT task). A WHRT task is a task that tolerates a precisely bounded number of deadline-misses in a sequence of k jobs[2].

Definition 3.6 (WHRT system). A WHRT system is a system that comprises at least one WHRT task.

Depending on the new constraints, we redefine the schedulability of a task.

[1] δ_i^-, δ_i^+ are the pseudo-inverse of $\tilde{\eta}_i^+$, $\tilde{\eta}_i^-$ as well.

[2] For the sake of brevity, we call such a sequence of k jobs a *k-sequence*.

Definition 3.7 ((m,k)-Schedulability). A WHRT task is said to be schedulable if it meets its (m, k) constraint.

TWCA applies to systems that are temporarily overloaded due to rare sporadic jobs, e.g., from interrupt service routines or recovery tasks [3]. In this paper, as in, e.g. [22], we assume that the task set is partitioned into so-called *overload tasks*, which are considered responsible for the overload situation, and *typical tasks*, which represent the nominal behavior of the system. The set of overload tasks (respectively typical tasks) is denoted $\mathcal{O}$ (respectively $\mathcal{T}$).

In this work, we assume that the scheduler is *deadline-miss agnostic*, i.e., it schedules tasks and lets them run to completion even if they have missed their deadline. Deadline-miss active schedulers (which may, e.g., kill the running job if it misses its deadline, or drop the next job/jobs) are out of the scope of this paper. Such schedulers can help schedule systems with utilization larger than 1, but are much more difficult to analyze and implement.

Definition 3.8. A *deadline miss model* (DMM) for a task τ_i is a function $dmm_i : \mathbb{N} \to \mathbb{N}$ such that out of any sequence of k consecutive jobs of τ_i, at most $dmm_i(k)$ may miss their deadline.

We can now formally define our problem: Given a task set $\mathcal{Z}$ defined as explained in this section, assuming it executes on a uniprocessor system scheduled with EDF scheduling, our objective is to compute a deadline miss model for each task $\tau_i \in \mathcal{T}$.

notation	description
$\mathcal{Z}$	Task set
$\mathcal{T}$	Typical task set
$\mathcal{O}$	Overload task set
U_i	Utilization of τ_i
U	Total utilization of the system
C_i	Worst-case execution time of τ_i
D_i	Relative deadline of τ_i
R_i^ℓ	Response time of job ℓ of τ_i
R_i^+	Worst-case response time of τ_i
d_i^ℓ	Absolute deadline of job ℓ of τ_i
$\mathbb{D}$	Set of absolute deadlines of tasks when they are released synchronously at $t = 0$
$\delta_i^-(n)$	Minimum distance between n jobs of τ_i
$\delta_i^+(n)$	Maximum distance between n jobs of τ_i
$\eta_i^+(\Delta)$	Maximum number of jobs of task τ_i within a right-open time interval Δ
$\tilde{\eta}_i^+(\Delta)$	Maximum number of jobs of task τ_i within a closed time interval Δ
L	Length of the synchronous busy-window in EDF
dbf_i	Demand bound function of τ_i
BW_i^+	Length of the worst-case level-i busy-window of τ_i in FPP
N_i	Upper bound on the number of deadline-misses that τ_i may miss within one busy-window
$\Omega_k^{s\to i}$	Upper bound on the number of sporadic jobs of τ_s that may interfere with any busy-window containing jobs of the k-sequence.
$\bar{c}$	Combination of sporadic overload tasks
$\bar{C}$	Set of unschedulable combinations
$dmm_i(k)$	Maximum number of deadline-misses that τ_i may miss out of k consecutive jobs

Table 1. Table of notations

[3]Note that the results presented in this paper directly apply to the analysis of communication resources by replacing the terms *task* and *processor* with *message* and *bus*, respectively.

TWCA relies on existing worst-case response time analyses as a foundation. Therefore, we call in the next section the worst-case response time analysis for EDF scheduling.

4 EDF RESPONSE TIME ANALYSIS

4.1 EDF scheduling policy

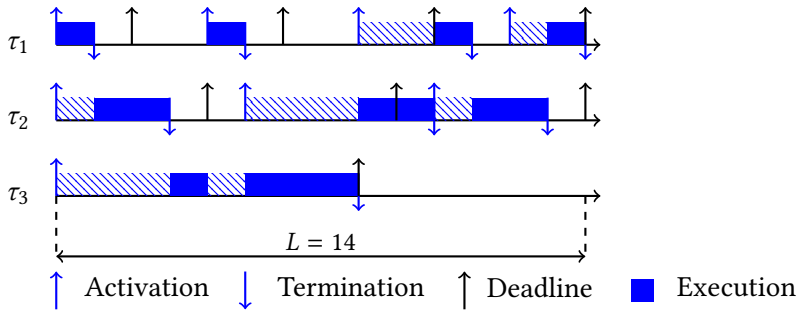

Fig. 1. The synchronous busy-window under EDF scheduling policy. The black upward arrow indicates D_i.

EDF is a dynamic priority scheduling algorithm, i.e., a task priority is changed regularly upon the strategy followed in the scheduling policy. In EDF, the task with the closest absolute deadline gets the resource. We consider a preemptive EDF scheduling, so the execution of a task may be interrupted by a job of another task with an earlier absolute deadline. In addition, EDF assigns the highest priority to the job with a missed deadline to complete execution. Therefore, it is a deadline-miss agnostic scheduler. Note that offset is not considered in this section.

4.2 Worst-case response time analysis

It is sufficient for hard real-time systems to test the schedulability with no need for computing the response time. In this case, a sufficient and necessary schedulability test can be satisfactory. In this work, we are interested in computing the response time and in the schedulability test as well. However, finding the worst-case response time of a task is not trivial when EDF scheduling is considered. We relay in this paper on Spuri's work presented in [42]. The exactness of Spuri's analysis has been preserved while the efficiency was the subject of later works, e.g., [6, 18, 43].

In this section, we show how to compute the worst-case response time R_i^+ for a given task τ_i and the length of the longest busy-window [42].

Definition 4.1. A *busy-window* $[t_1, t_2[$ is a time interval such that for any $t \in [t_1, t_2[$, the resource is busy (i.e., some task is scheduled) at t.

Of all the possible busy-windows, the *synchronous busy-window* plays a key role for response time analysis, see Figure 1.

Definition 4.2. The *synchronous busy-window* for $\mathcal{Z}$ is a busy-window $[t_1, t_2[$ in which all tasks are activated synchronously at t_1 and then according to the maximum arrival function η^+.

The synchronous busy-window has the following important property.

Lemma 4.3 ([42]). *The length L of the synchronous busy-window is the maximum length of any possible busy-window in any schedule.*

Proof. This proof sketch depends on the argumentation presented in A.2 in [42]. Any busy-window rather than the synchronous one does not satisfy one or both following conditions:

(1) all tasks are activated synchronously at the beginning of the busy-window.
(2) the arrival of activations follows the maximum arrival function η^+.

However, shifting left all jobs in the busy-window in order to satisfy the above two conditions can only bring more workload to be processed during the busy-window which makes it longer. The synchronous busy-window satisfies both above conditions, and it has therefore the maximum length. □

A second property of the synchronous busy-window that is needed for response time analysis is in the following lemma.

Lemma 4.4 ([42]). *The worst-case response time of a task τ_i is found in a busy-window in which all tasks other than τ_i are released synchronously at the beginning of the busy-window and then at their maximum rate.*

Proof. This lemma is lemma 4.1 in [42]. The key argument to prove this lemma is that if all jobs of tasks different from τ_i are "shifted left" such that they are released synchronously at the beginning of the busy-window, the workload of jobs of τ_i cannot diminish. □

To compute the response time of a job ℓ of τ_i activated at $\mathfrak{a}$ with absolute deadline d_i^ℓ, it is sufficient to be aware of a part of the busy-window in which only jobs with absolute deadline smaller than or equal to d_i^ℓ execute. This part of the busy-window relative to the absolute deadline d_i^ℓ is named *deadline-d busy-window* and $L_i(\mathfrak{a})$ denotes its length.

We are looking for a job ℓ activated at time $\mathfrak{a} \geqslant 0$ such that R_i^ℓ is the worst-case response time: $R_i^+ = R_i^\ell$. Let $\alpha_i^0(\mathfrak{a})$ denote the activation time of the first job, which has the order 0, of τ_i in a busy-window where the job ℓ activated at $\mathfrak{a}$ belongs to the same busy-window.

$$\alpha_i^0(\mathfrak{a}) = \mathfrak{a} - \delta_i^-(\eta_i^+(\mathfrak{a})) \tag{3}$$

Lemma 4.3 shows that L is the maximum length of any busy-window, therefore, the significant values of $\mathfrak{a}$ are in the interval $[0, L - C_i[$. Furthermore, in [42] the author claims that it is not difficult to see that the local maxima of $L_i(\mathfrak{a})$ is found for those values of $\mathfrak{a}$ such that in the arrival pattern there is at least an instant of a task different from τ_i with deadline equal to d_i^ℓ, or all tasks are synchronized, i.e., $\alpha_i^0(\mathfrak{a}) = 0$. This claim can be proven using the argument given to prove Lemma 4.4.

Lemma 4.5. *The response time of a job ℓ of τ_i with absolute deadline d_i^ℓ does not decrease if all jobs of τ_i are "shifted" to the left such that d_i^ℓ coincides with the immediate previous absolute deadline, let d_p denote it, of a job of a task different from τ_i.*

Proof. Shifting left keeps the priority of the job ℓ without changing as ℓ is shifted such that d_i^ℓ coincides the closest previous higher priority absolute deadline d_p. That will not decrease the workload that interferes with ℓ. By comparing Figure 2.a with Figure 2.b we observe that the response time of τ_i does not decrease.

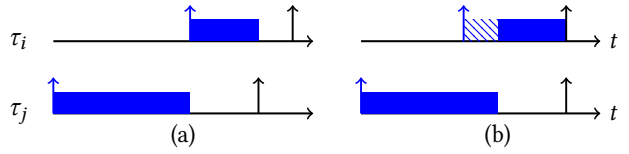

Fig. 2. The impact of shifting shifting left on the response time of τ_i

$task$	C_i	D_i	T_i
τ_1	1	2	4
τ_2	2	4	5
τ_3	4	8	15

$\mathfrak{a}$	R_3^ℓ
0	8
1	9
2	9
6	9

Table 2. The worst-case response time analysis of τ_3.

□

Lemma 4.5 shows that it is sufficient to say:

$$R_i^+ = \max_{\substack{\mathfrak{a}=\mathfrak{d}-D_i \\ \mathfrak{d}\in\mathbb{D}}} \{R_i^\ell\}$$

where R_i^ℓ is the response time of the job ℓ activated at $\mathfrak{a}$, and $\mathbb{D}$ is the set of absolute deadlines of tasks when they are released synchronously at $t = 0$:

$$\mathbb{D} = \cup_{j\in\mathcal{Z}}\{d^l | d^l = \delta_j^-(l) + D_j \wedge d^l < L, l \in \mathbb{N}^*\} \tag{4}$$

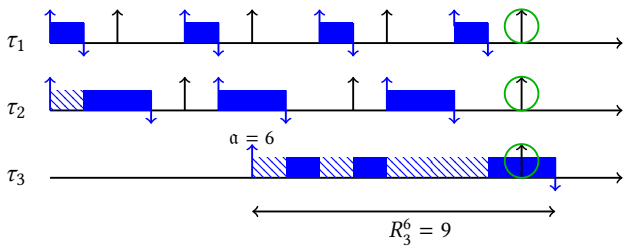

Fig. 3. The worst-case response time of τ_3. The job is activated at $\mathfrak{a} = 9$.

Example 4.6. Let us consider the task set shown in Table 2. We want to compute the worst-case response time of τ_3. In this system $L = 14$ as Figure 1 shows. The set $\mathbb{D}$ is: $\mathbb{D} = \{9, 10, 14\}$. The equivalent values of $\mathfrak{a}$ are shown in Table 2. Figure 3 illustrates the worst-case response time.

4.3 Algorithm to compute worst-case response time

We start with bounding the length of the synchronous busy-window L (see Definition 4.2):

$$\begin{cases} L^{(0)} = \sum_{j\in\mathcal{Z}} C_j \\ L^{(m+1)} = \sum_{j\in\mathcal{Z}} \eta_j^+(L^{(m)}) \cdot C_j \end{cases} \tag{5}$$

where the equation is equivalent to Equation 1 in [42] and it converges when $L^{(m+1)} = L^{(m)}$.

After bounding L, we can compute $\mathbb{D}$ which represents the set of candidates to bound the worst-case response time, Equation 4.

The next step is to bound the contribution of tasks in the deadline busy-window $L_i(\mathfrak{a})$. Assume that the beginning of the busy-window is at $t_0 = 0$. Up to time t, $\eta_j^+(\Delta = t - 0)$ jobs of τ_j will have been released for $j \neq i$ and there will be no more than $\tilde{\eta}_j^+(\mathfrak{a} + D_i - D_j)$ jobs contribute in $L_i(\mathfrak{a})$.

$$W_i(\mathfrak{a}, t) = \sum_{\substack{j\neq i \\ D_j \leqslant \mathfrak{a}+D_i}} \min\{\eta_j^+(t), \tilde{\eta}_j^+(\mathfrak{a} + D_i - D_j)\} \cdot C_j + \lambda_i(\mathfrak{a}, t) \cdot C_i \tag{6}$$

Where $\lambda_i(\mathfrak{a}, t)$ is the number of jobs of τ_i that are activated in $[0, t[$:

$$\lambda_i(\mathfrak{a}, t) = \begin{cases} \min\{\eta_i^+(t - \alpha_i^0(\mathfrak{a})), \tilde{\eta}_j^+(\mathfrak{a})\} & t > \alpha_i^0(\mathfrak{a}), \\ 0 & otherwise. \end{cases} \tag{7}$$

Then, the deadline busy-window $L_i(\mathfrak{a})$ is bounded as follows:

$$\begin{cases} L_i^{(0)}(\mathfrak{a}) = \sum_{\substack{j \neq i \\ D_j \leqslant \mathfrak{a} + D_i}} C_j + I_{\{\alpha_i^0(\mathfrak{a})=0\}} \cdot C_i \\ L_i^{(m+1)}(\mathfrak{a}) = W_i(\mathfrak{a}, L_i^{(m)}(\mathfrak{a})) \end{cases} \tag{8}$$

Where $I_{\{\alpha_i^0(\mathfrak{a})=0\}}$ is a Boolean such that:

$$I_{\{\alpha_i^0(\mathfrak{a})=0\}} = \begin{cases} 1 & \alpha_i^0(\mathfrak{a}) = 0, \\ 0 & otherwise. \end{cases} \tag{9}$$

Note that Equation 8 is equivalent to Equation 3 in [42].

The response time of the job ℓ that is activated at $\mathfrak{a}$ is then:

$$R_i^\ell = \max\{L_i(\mathfrak{a}) - \mathfrak{a}, C_i\} \tag{10}$$

The computation of the worst-case response time of a given task τ_i is shown in Algorithm 1.

Algorithm 1: Compute the worst-case response time

```
Compute L (Equation 5);
for 𝔡 ∈ 𝔻 do
    a = 𝔡 − D_i;
    α_i^0(a) = a − δ_i^−(η_i^+(a));
    Compute L_i(a) (Equation 8);
    Compute R_i^ℓ (Equation 10);
R_i^+ = max{R_i^ℓ}
```

4.4 Sufficient and necessary schedulability test

For EDF scheduling, a sufficient and necessary schedulability test has been developed based on the *demand bound function*.

Definition 4.7. Demand bound function dbf_i is defined as follows:

$$dbf_i(t) := \tilde{\eta}_i^+(t - D_i) \cdot C_i \tag{11}$$

Theorem 4.8. *A task set $\mathcal{Z}$ is schedulable if and only if:*

$$\forall t \in \mathbb{D} : \sum_{\substack{\forall i \in \mathcal{Z} \\ D_i \leqslant t}} dbf_i(t) \leqslant t \tag{12}$$

Proof. It has been proven in Theorem 6 in [29] and Theorem 3.1 in [42] that if a deadline-miss can occur in a busy-window, then a deadline-miss occurs in the synchronous busy-window. Therefore, it is sufficient to test the schedulability in the synchronous busy-window.

At each point of time $t \in \mathbb{D}$ if the demanded workload to be processed up to t is less than t, then the absolute deadline at t will be guaranteed. Thus, it is necessary to test the schedulability $\forall t \in \mathbb{D}$. □

5 STATE-OF-THE-ART TWCA

TWCA is proposed in [20, 45] to provide WHRT guarantees in the form of a deadline miss model. TWCA considers systems that are temporarily overloaded due to rare sporadic jobs, e.g., from interrupt service routines or recovery tasks. In this paper, as in, e.g. [22], we assume that the task set $\mathcal{Z}$ is partitioned into overload tasks $\mathcal{O}$ and typical tasks $\mathcal{T}$, which represent the nominal behavior of the system. TWCA relies on the assumption that a typical task $\tau_i \in \mathcal{T}$ may miss its deadline only with the presence of *any* overload task $\tau_s \in \mathcal{O}$, otherwise, τ_i is schedulable. Thus, TWCA does not apply to systems in which a typical task is not schedulable with the absence of all overload tasks.

In this section, we show how to compute DMMs for FPP scheduling. In TWCA, the concept of *busy-window* has a crucial rule in computing DMMs. In FPP, a level-i busy-window is a maximal time interval $[t_1, t_2[$ during which the processor is busy w.r.t. task τ_i such that jobs of tasks of equal or higher priority than τ_i are still pending. The longest such interval, called worst-case level-i busy-window and its length is denoted by BW_i^+, is built such that τ_i and higher-priority tasks are released synchronously at the beginning of the busy-window and then as early as possible. Figure 4 illustrates the worst-case level-i busy-window.

Let $hp(i)$ denotes the set of tasks with higher priority or equal to the priority of τ_i. In FPP, if a job ℓ of $\tau_j \in hp(i)$ is activated within a level-i busy-window $[t_1, t_2[$, i.e., $act_j^\ell \geqslant t_1$, then it completes execution in the same level-i busy-window $end_j^\ell \leqslant t_2$. Following this property of the level-i busy-window in FPP, we deduce that τ_i misses its deadline in the level-i busy-window due to sporadic overload jobs that are activated in the same level-i busy-window. Using this observation, TWCA computes $dmm_i(k)$ by:

- bounding the number of deadline-misses that τ_i may miss within one level-i busy-window; let N_i denote this bound.
- bounding the number of level-i busy-windows in which jobs of the k-sequence miss their deadline; let NBW_i^{miss} denotes this bound.

Then, $dmmi(k)$ can be bounded safely as

$$dmmi(k) := N_i \,.\, NBW_i^{miss} \tag{13}$$

Definition 5.1. N_i is an upper bound on the number of deadlines that τ_i may miss within any level-i busy-window.

In FPP, N_i is computed as follows:

$$N_i = \#\{q \mid 1 \leqslant q \leqslant Q_i : R_i^q > D_i\} \tag{14}$$

Where Q_i denotes the maximum number of jobs of τ_i in the worst-case level-i busy-window.

Bounding $NBWi^{miss}$ is more complex. We bound first the number of sporadic overload jobs.

Definition 5.2. $\Omega_k^{s\to i}$ is an upper bound on the number of sporadic overload jobs of $\tau_s \in \mathcal{O}$ that may interfere with any level-i busy-window containing jobs of the k-sequence.

In FPP, $\Omega_k^{s\to i}$ is computed as follows:

$$\Omega_k^{s\to i} = \eta_s^+(BW_i^+ + \delta_i^+(k) + R_i^+) \tag{15}$$

The k-sequence is represented by $\delta_i^+(k)$ which is the maximum distance between k jobs. Note that any sporadic overload job is activated before the level-i busy-window of the first job of the k-sequence will not interfere with its execution. On the other hand, the last job of the k-sequence will not be impacted by any overload job that is activated after its completion time.

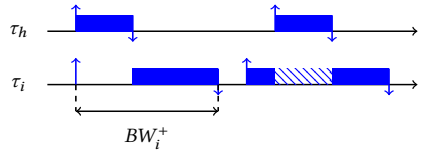

Fig. 4. FPP scheduling where τ_h has a higher priority than τ_i

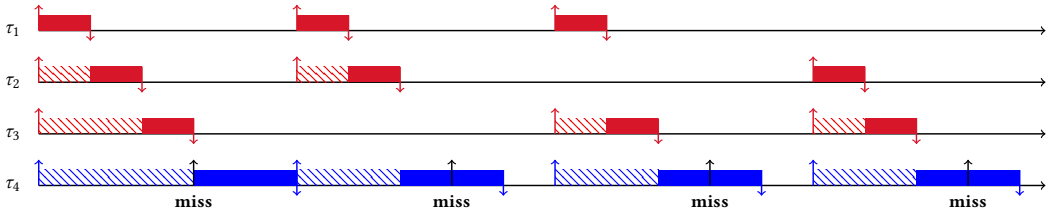

Fig. 5. Illustration of the unschedulable combinations $\bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4$ in Example 5.5 where FPP is considered. Note that $N_i = 1$ in this figure.

By assuming that one overload job alone is sufficient to cause N_i deadline-misses, we can then safely compute DMM as

$$dmm_i(k) = N_i \cdot \sum_{s \in O} \Omega_k^{s \to i} \tag{16}$$

Such bound is too pessimistic as one sporadic overload job alone may not cause a deadline-miss. Clearly, there is room to tight this bound by considering a combination of sporadic overload tasks which are activated within one level-i busy-window and lead to deadline-misses.

Definition 5.3. A combination of sporadic tasks $\bar{c}$ is a subset of overload tasks: $\bar{c} \subseteq O$.

Let n_s denotes number of overload tasks, i.e., $n_s = \#\{O\}$. There are $2^{n_s} - 1$ combinations. By $R_i^{\bar{c}}$ we denote the worst possible response time of τ_i in the task set $\mathcal{T} \cup \bar{c}$. In this way $R_i^+ = R_i^{\mathcal{Z}}$ where $\mathcal{Z} = \mathcal{T} \cup O$.

Definition 5.4. We say that a combination $\bar{c} \subseteq O$ is unschedulable w.r.t τ_i if $R_i^{\bar{c}} > D_i$.

In the same way we call the combination $\bar{c}$ for which $R_i^{\bar{c}} \leqslant D_i$ a schedulable combination.

In the worst-case, the available sporadic overload jobs $\sum_{s \in O} \Omega_k^{s \to i}$ will form a finite number of instances $x_{\bar{c}}$ of each unschedulable combination $\bar{c}$ such that $0 \leqslant x_{\bar{c}} \leqslant \min_{s \in \bar{c}}\{\Omega_k^{s \to i}\}$. That means, $\sum_{s \in O} \Omega_k^{s \to i}$ may screw up at most $\sum_{\bar{c} \in \tilde{C}} x_{\bar{c}}$ level-i busy-windows where $\tilde{C}$ denotes the set of unschedulable combinations. NBW_i^{miss} can be bounded as follows:

$$NBW_i^{miss} \leqslant \max\Big\{\sum_{\bar{c} \in \tilde{C}} x_{\bar{c}}\Big\} \tag{17}$$

Conservatively, we can compute a DMM for any task $\tau_i \in \mathcal{T}$ using the following ILP:

$$dmm_i(k) = \max\Big\{N_i \sum_{\bar{c} \in \tilde{C}} x_{\bar{c}} \ : \ \sum_{\bar{c}: s \in \bar{c} \in \tilde{C}} x_{\bar{c}} \leqslant \Omega_k^{s \to i} \ \forall s \in O, x_{\bar{c}} \in \mathbb{N} \ \forall \bar{c} \in \tilde{C}\Big\} \tag{18}$$

Example 5.5. Consider a system with 4 tasks where τ_1, τ_2, τ_3 are sporadic overload tasks with $\Omega_k^{1 \to 4} = 2$, $\Omega_k^{2 \to 4} = 2$ and $\Omega_k^{3 \to 4} = 2$ where task τ_4 has a lower priority than the sporadic tasks.

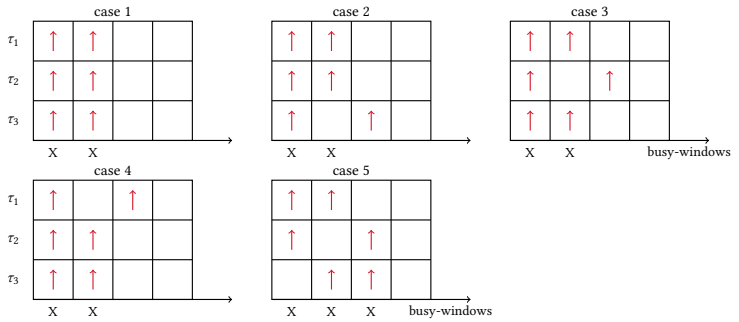

Fig. 6. Packing overload jobs into busy-windows of τ_4 where X = deadline-miss. In this example ,Example 5.5, there are 5 unschedulable cases.

With 3 overload tasks we get 7 combinations, assume that the set of unschedulable combinations is $\tilde{C} = \{\bar{c}_1 = \{\tau_1, \tau_2, \tau_3\}, \bar{c}_2 = \{\tau_1, \tau_2\}, \bar{c}_3 = \{\tau_1, \tau_3\}, \bar{c}_4 = \{\tau_2, \tau_3\}\}$ and the other 3 combinations are schedulable. Figure 5 shows the unschedulable combinations. Using the given $\Omega_k^{s\to i}$ values we can combine unschedulable combinations in 5 different ways:

(1) $x_{\bar{c}_1} = 2 \Rightarrow \sum_{\bar{c}\in\tilde{C}} x_{\bar{c}} = 2$
(2) $x_{\bar{c}_1} = 1, x_{\bar{c}_2} = 1 \Rightarrow \sum_{\bar{c}\in\tilde{C}} x_{\bar{c}} = 2$
(3) $x_{\bar{c}_1} = 1, x_{\bar{c}_3} = 1 \Rightarrow \sum_{\bar{c}\in\tilde{C}} x_{\bar{c}} = 2$
(4) $x_{\bar{c}_1} = 1, x_{\bar{c}_4} = 1 \Rightarrow \sum_{\bar{c}\in\tilde{C}} x_{\bar{c}} = 2$
(5) $x_{\bar{c}_2} = 1, x_{\bar{c}_3} = 1, x_{\bar{c}_4} = 1 \Rightarrow \sum_{\bar{c}\in\tilde{C}} x_{\bar{c}} = 3$

Then $dmm_i(k) = 3 \times N_i$ is the maximum number of deadline-misses that τ_4 may miss in the k-sequence. Figure 6 illustrate the 5 unschedulable cases.

Note that the above ILP is not only NP-hard [33] but also it might have an exponential size input. Hence, directly computing $dmm_i(k)$ is impractical, instead, we consider the LP relaxation in [45]. The linear program of (18) is:

$$dmm'_i(k) = \max\left\{N_i \sum_{\bar{c}\in\tilde{C}} x_{\bar{c}} \;:\; \sum_{\bar{c}:s\in\bar{c}\in\tilde{C}} x_{\bar{c}} \leqslant \Omega_k^{s\to i}\ \forall s \in O, x_{\bar{c}} \geqslant 0\ \forall \bar{c} \in \tilde{C}\right\} \tag{19}$$

The dual linear program of (19) reads as follows:

$$\min\left\{\sum_{s\in O} \Omega_k^{s\to i} y_s \;:\; \sum_{s\in\bar{c}, s\in O} y_s \geqslant N_i\ \forall \bar{c} \in \tilde{C}, y_s \geqslant 0\ \forall s \in O\right\} \tag{20}$$

We proposed in [45] an algorithm using *column generation*. In the generating columns based algorithm, we start with a reduced set of unschedulable combinations to a smaller sample $W \subseteq \tilde{C}$. Initially, this could be $W := O$ (O is guaranteed to be unschedulable). Let x^* be an optimal primal solution, and $y$fk be an optimal dual solution for the reduced LP with objective function z^*. We aim to find a violated constraint of the complete dual LP to identify the next variable that will be added to the LP, which indicates the next unschedulable combination that will be added to W, by computing

$$v = \min_{\bar{c}\in\tilde{C}_i} \sum_{s\in\bar{c}, s\in O} y_s^*. \tag{21}$$

When there are no more variables that violate the constraint of the complete dual LP, i.e., $v \geqslant N_i$, the column generation finishes, and we find a globally optimal solution $z^* = dmm'_i(k) \geqslant dmm_i(k)$. As

this might take a long time due to the exponential size of $\tilde{C}_i$, constructing a DMM from suboptimal solutions can be sufficient. For this end, we define

$$dmm_i''(k) := \frac{N_i}{\nu} z^* \tag{22}$$

If $0 < \nu \leqslant N_i$, then $dmm_i''(k) \geqslant dmm_i'(k)$. Therefore, $dmm_i''(k)$ is a DMM. Proofs and further discussion are available in [45].

6 COMPUTING DMM FOR EDF SCHEDULING

In this section, we show how to compute WHRT guarantees for temporarily overloaded systems that are scheduled with EDF. To do so, we extend TWCA [20, 45] to systems with EDF scheduling.

The notation of busy-window as defined in Definition 4.1 satisfies that if a job ℓ of $\tau_j \in \mathcal{Z}$ is activated within a busy-window $[t_1, t_2[$, i.e., $act_j^\ell \geqslant t_1$, then it completes execution in the same busy-window $end_j^\ell \leqslant t_2$. Therefore, we can apply TWCA to EDF scheduling. To compute a DMM for a given task $\tau_i \in \mathcal{T}$ using TWCA:

(1) we compute an upper bound N_i on the number of deadlines that τ_i may miss within one busy-window.
(2) we compute an upper bound $\Omega_k^{s \to i}$ on the number of overload jobs of τ_s that may interfere with any busy-window containing jobs of the k-sequence.
(3) we propose a criterion to compute the unschedulable combinations efficiently.

Compute N_i. The impact of sporadic overload jobs is enclosed in the busy-window during which they have been executed. Hence, the number of deadlines that τ_i may miss within one busy-window due to one or more sporadic overload jobs can be bounded.

Lemma 6.1. *For a given task τ_i, let $\mathbb{BW}_i$ indicates the set of busy-windows that each of which satisfies: all tasks but τ_i are activated synchronously at the beginning of the busy-window and then at their maximum rate; and the first job of τ_i in the busy-window is activated at $\mathfrak{a}$ where $\mathfrak{a} = \mathfrak{d} - D_i : \mathfrak{d} \in \mathbb{D}$. Also, $\forall b \in \mathbb{BW}_i$ let n_b indicates the number of deadline-misses within the busy-window b. Let*

$$N_i = \max_{\forall b \in \mathbb{BW}_i} \{n_b\} \tag{23}$$

Then N_i is an upper bound on the number of deadlines that τ_i may miss within one busy-window.

Proof. It has been proven in [42] that within a busy-window in which all tasks but τ_i are activated synchronously at the beginning of the busy-window, and then at their maximum rate, the workload of all other tasks cannot diminish and the response time of jobs of τ_i within this busy-window can only increase. A certain distribution of the interfering workload will cause the maximum number of deadline-misses to τ_i within the busy-window. Such a busy-window is not necessary to be the synchronous one, in which $\mathfrak{a} = 0$. Therefore, all possible busy-window candidates have to be checked. □

Compute $\Omega_k^{s \to i}$.

Lemma 6.2. *$\Omega_k^{s \to i}$ is computed as follows:*

$$\Omega_k^{s \to i} = \tilde{\eta}_s^+(L + \delta_i^+(k) + \max\{D_i - D_s, 0\}) \tag{24}$$

Proof. Figure 7 illustrates $\Omega_k^{s \to i}$. We know that $\tilde{\eta}_s^+(\Delta)$ returns the maximum number of jobs that may be activated within a closed time interval. We should carefully bound the window of k-sequence by considering the maximum distance between k jobs and sufficient time windows

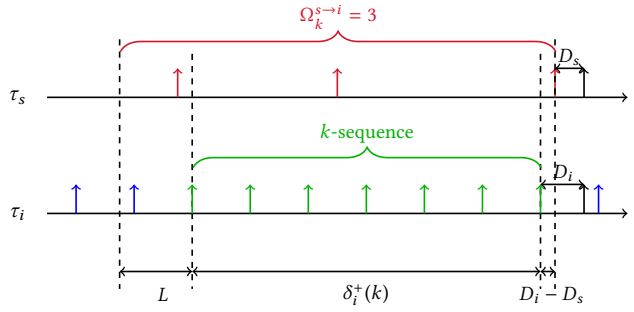

Fig. 7. $\Omega_k^{s\to i}$ under EDF scheduling policy.

before and after it during which the execution of the first and the last jobs of the k-sequence might be impacted.

$\delta_i^+(k)$: the longest closed time interval that contains k consecutive jobs is bounded by $\delta_i^+(k)$. An overload job that occurs during it may have an impact on the response times of the k-sequence.

L: An overload job will have no impact on the first job of the k-sequence unless they belong to the same busy-window. L is proven to be the maximum length of a busy-window.

$\max\{D_i - D_s, 0\}$: Any sporadic overload job with an absolute deadline beyond the absolute deadline of the k-th job has no impact on any job belongs to the k-sequence. □

Schedulability criterion. A task τ_i misses its deadline when it experiences interfering from unschedulable combinations, see Definition 5.4. To compute DMMs using TWCA, the set of unschedulable combinations has to be computed. In EDF scheduling policy, a sufficient and necessary schedulability test based on the demand bound function is used [42]. The demand bound function is defined in Definition 4.7, and the schedulability test is presented in Equation 12. Based on this schedulability test we compute the set of unschedulable combinations as follows:

Lemma 6.3. *A combination $\bar{c}$ is unschedulable if (necessary condition) the task set $\mathcal{T} \cup \bar{c}$ is not schedulable:*

$$\forall t \in \mathbb{D} : \sum_{\substack{\forall i \in \{\mathcal{T} \cup \bar{c}\} \\ D_i \leqslant t}} dbf_i(t) > t \tag{25}$$

Proof. If the system is not schedulable with the presence of the combination $\bar{c}$, it does not mean that τ_i misses its deadline. In other words, the set of unschedulable combinations w.r.t. τ_i is a subset of the set of unschedulable combinations generated using the above condition. Therefore, the above condition is a necessary schedulability criterion w.r.t. τ_i. □

The superset of unschedulable combinations w.r.t. τ_i is then:

$$\tilde{C}_i = \{\bar{c} \mid \exists t \in \mathbb{D} : \sum_{\substack{\forall i \in \{\mathcal{T} \cup \bar{c}\} \\ D_i \leqslant t}} dbf_i(t) > t\} \tag{26}$$

Note that for EDF scheduling $\forall i, j \in \mathcal{T} : \tilde{C}_i = \tilde{C}_j$ because $\tilde{C}$ is computed regardless the considered task, see Lemma 6.3.

Compute DMM. Each unschedulable combination may impact at most one busy-window causing at most N_i deadline-misses. N_i is a constant, however, the number of impacted busy-windows varies depending on the way we combine the available sporadic overload jobs $\sum_{s \in O} \Omega_k^{s\to i}$ in unschedulable

combinations. To conservatively compute the DMM, we need to maximize the number of impacted busy-windows, thus, the following theorem:

Theorem 6.4 ([45]). *We compute a DMM for any task $\tau_i \in \mathcal{T}$ as follows:*

$$dmm_i(k) = \max \left\{ N_i \sum_{\bar{c} \in \tilde{C}_i} x_{\bar{c}} \; : \; \sum_{\bar{c}: s \in \bar{c} \in \tilde{C}_i} x_{\bar{c}} \leqslant \Omega_k^{s \to i} \; \forall s \in O, x_{\bar{c}} \in \mathbb{N} \; \forall \bar{c} \in \tilde{C}_i \right\} \tag{27}$$

where $x_{\bar{c}}$ represents the number of instances of the unschedulable combination $\bar{c}$.

Proof. On the one hand, within one busy-window there will be no more than N_i deadline-misses as has been proven in Lemma 6.1.

On the other hand, there will be no more than $\max\{\sum_{\bar{c} \in \tilde{C}_i} x_{\bar{c}}\}$ impacted busy-windows (each includes at least one deadline-miss).

The ILP in Equation 27 provides, therefore, an upper bound on the number of deadlines that τ_i might miss out of any k-sequence, i.e., a DMM.

□

Note that the ILP does not depend on the scheduling policy. What changes w.r.t. the considered scheduling policy is how N_i, $\tilde{C}_i$, and $\Omega_k^{s \to i}$ are computed.

This ILP can compute DMMs for tasks for which arrival curves are used to model task activations. Hence, it is more general than that in [4, 44]. Nevertheless, the proposed work is not able to compute DMMs for tasks that have $\delta^+(2) = \infty$, i.e., sporadic tasks, because we will not be able to bound $\Omega_k^{s \to i}$. In the next section, we extend this discussion by addressing the sources of pessimism in computing DMMs.

7 SOURCES OF PESSIMISM

Computing $dmm_i(k)$ using the proposed analysis has three main sources of pessimism:

(1) N_i is only an upper bound on the number of deadlines that τ_i may miss within one busy-window. Not every unschedulable combination causes the same number of deadline-misses. Figure 8 illustrates this case. While the combination $\bar{c}_1 = \{\tau_1, \tau_2\}$ causes two deadline-misses and therefore $N_i = 2$, $\bar{c}_2 = \{\tau_2\}$ cannot cause more than one deadline-miss. When $N_i = 1$, there is no pessimism related to N_i.

(2) TWCA assumes that every busy-window within the time window of the k-sequence experiences the maximum interference from all tasks $\tau_i \in \mathcal{T}$. This may not be possible given the activation models of tasks in $\mathcal{T}$ within the time window of the k-sequence. Consequently, the maximum number of impacted busy-windows are over-approximated.
Figure 9 shows a scenario in which τ_4 meets its deadline even with the presence of the *unschedulable combinations* $\bar{c}_2 = \{\tau_2\}$. Over a k-sequence, two busy-windows that both experience $\bar{c}_2$ cannot both miss deadlines because they cannot both be interfered by τ_3.

(3) When a sporadic task τ_s has more than one job within the longest busy-window. Let ω_s denotes the number of jobs of τ_s within the longest busy-window. In a combination, each task τ_s has a Boolean representation: either ω_s jobs are activated as early as possible according to the activation model of τ_s when $\tau_s \in \bar{c}$ or none of them when $\tau_s \notin \bar{c}$.

In the next section, we motivate and validate our analysis a realistic case study and a set of synthetic test cases to extensively test our analysis.

8 EXPERIMENTS

Two kinds of experiments will be presented in this section: a realistic case study inspired by industrial practice, and synthetic test cases.

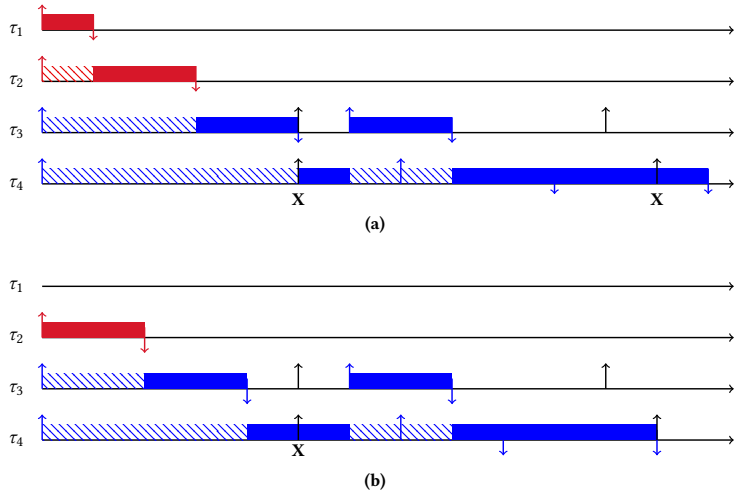

Fig. 8. N_i as a source of pessimism in the computations of $dmm_i(k)$. The black upward arrows indicate the deadline. **X** indicates a deadline-miss.

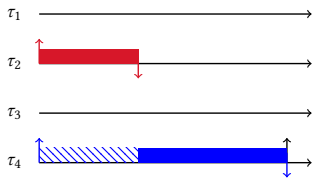

Fig. 9. Second source of pessimism in the computation of $dmm_i(k)$. When τ_3 does not interfere with τ_4, then the combination $\bar{c}_2 = \{\tau_2\}$ is not unschedulable any more.

For these experiments, the worst-case response time analysis of EDF scheduling is implemented and integrated with pyCPA [15], which is a python implementation of Compositional Performance Analysis (CPA)[23]. We use pyCPA as a core analysis for our experiments.

Remember that we consider the LP relaxation in [45] to compute $dmm_i(k)$ instead of directly computing it using the ILP in Theorem 6.4. The optimization problem is solved using cplex.

Concerning the schedulability criterion in Equation 26, we implement the Quick convergence Processor-demand Analysis (QPA) algorithm [46] that reduces significantly the number of iterations required to check the schedulability of a given task set.

8.1 Satellite on-board software

We start with a case study inspired by industrial practice, which is provided by Thales Alenia Space (TAS) and published in [22]. The case study is a satellite on-board software.

A satellite comprises two major parts: the platform and the payload. While the payload's software has soft real-time requirements, the platform functions, e.g., the Attitude and Orbit Control System (AOCS) and the Thermal Control System (TCS), are characterized by hard real-time requirements. However, some functions of the platform on-board software (OBSW) may occasionally miss few deadlines without jeopardizing the mission which motivates considering WHRT requirements instead of hard ones.

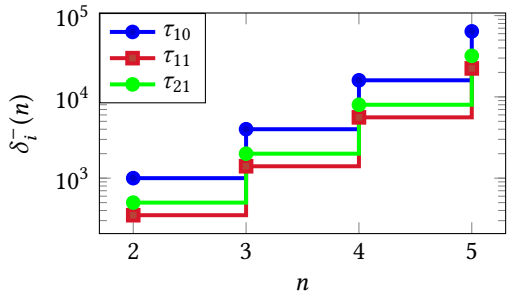

Fig. 10. Activation models for τ_{10}, τ_{11} and τ_{21}.

In this experiment, we recall the case study from [22] but with considering EDF scheduling instead of fixed priority scheduling. The task set representative of OBSW consists of 30 tasks: 27 tasks are in the nominal mode and 3 recovery and reconfiguration tasks (τ_{10}, τ_{11} and τ_{21}) which cause a transient overload on the system when they are activated. Table 3 brings the task set from Table 1 in [22] and from the results in Section 8.1 where τ_{10}, τ_{11} and τ_{21} appear in red. Activation models for τ_{10}, τ_{11} and τ_{21} are illustrated in Figure 10.

When considering EDF scheduling, **no task misses its deadline**. For the sake of the case study, we shorten the deadline of the sporadic overload tasks $\tau_{10}, \tau_{11}, \tau_{21}$ such that $\forall s \in \{10, 11, 21\} : D_s = 8 * C_s$ (they are not shown in Table 3). In this case, 11 nominal tasks miss their deadline with the presence of the sporadic overload, namely: $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_9, \tau_{12}, \tau_{13}$ and τ_{16}.

Table 4 shows $dmm_i(k) : k = 2, 10, 100, 500, 1000$ for the impacted tasks. Not surprisingly, τ_1, τ_2 have the worst DMMs as they have the shortest periods. Because of the inborn robustness of the implemented control laws, OBSW tasks might tolerate few deadline-misses. That is, tasks $\tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_{12}$ need to tolerate only 1 deadline-miss out of any sequence of 10 deadlines while τ_9, τ_{13} and τ_{16} need to tolerate no more than 1 deadline-miss in a row and 3 out of any consecutive 10 deadlines. However, τ_1 and τ_2 may miss up to 3 consecutive deadlines when the 3 recovery and reconfiguration tasks are activated within one busy-window which could be intolerable and lead to a functional error.

Let us now repeat the experiment for FPP scheduling in order to illustrate the impact of EDF scheduling on the DMMs of real-time tasks by comparing the results of the two schedulers (EDF, FPP). Note that for FPP, τ_1 has the highest priority while τ_{30} has the lowest one and the index of each task refers to its priority. The task set has a criticality-based priority assignment; neither Rate Monotonic (RM) nor Deadline Monotonic (DM) is used.

In FPP scheduling, only three nominal tasks miss their deadline as shown in Table 5. As the priority is fixed, we can predict which task may miss its deadline. In this case study nominal tasks τ_1 to τ_9 will never be impacted by the sporadic overload tasks ($\tau_{10}, \tau_{11}, \tau_{21}$), while the rest might be impacted and miss few deadlines when the 3 recovery and reconfiguration tasks are activated within one level-i busy-window. In this case, TWCA helps in bounding the distribution of the deadline-misses/hits in the form of a DMM.

EDF scheduling sacrifices no task, instead, it fairly distributes the deadline-misses. Therefore, it is not possible to predict which task will miss its deadline. EDF is therefore can not guarantee the schedulability of systems that have hard and WHRT tasks.

In the next experiment, we present a comparison between DMMs computed under EDF and RM using synthetic test cases to study the impact of EDF scheduling on DMMs extensively.

Name	C	T	D	**Name**	C	T	D
τ_1	0.56	15.625	15.625	τ_{16}	3.5	250	250
τ_2	0.76	15.625	15.625	τ_{17}	27	500	500
τ_3	15	125	31.25	τ_{18}	1.5	1000	1000
τ_4	25.03	125	46.875	τ_{19}	16	1000	1000
τ_5	7.5	62.5	62.5	τ_{20}	19.1	1000	1000
τ_6	6.15	125	125	τ_{21}	36.02	500	1000
τ_7	1.2	125	125	τ_{22}	88.8	2000	2000
τ_8	0.9	1000	500	τ_{23}	2	32000	32000
τ_9	1.95	250	250	τ_{24}	1	32000	32000
τ_{10}	30	10000	125	τ_{25}	1	1000	1000
τ_{11}	30	350	125	τ_{26}	20	1000	1000
τ_{12}	1.2	125	125	τ_{27}	40	2000	2000
τ_{13}	5.15	250	203.125	τ_{28}	1.5	2000	2000
τ_{14}	1.2	1000	500	τ_{29}	1.5	2000	2000
τ_{15}	22.5	500	500	τ_{30}	0.2	32000	32000

Table 3. OBSW tasks. C, T and D denote respectively: worst-case execution time, period/minimum distance, deadline. The time unit is ms.

Name	$dmm_i(2)$	$dmm_i(10)$	$dmm_i(100)$	$dmm_i(500)$	$dmm_i(1000)$
τ_1	2	3	3	9	12
τ_2	2	3	3	9	12
τ_3	1	1	3	4	4
τ_4	1	1	3	4	4
τ_5	1	1	3	4	4
τ_6	1	1	3	4	4
τ_7	1	1	3	4	4
τ_9	1	3	4	4	4
τ_{12}	1	1	3	4	4
τ_{13}	1	2	4	4	4
τ_{16}	1	3	4	4	4

Table 4. DMMs for OBSW tasks when EDF scheduling is considered.

Name	$dmm_i(2)$	$dmm_i(10)$	$dmm_i(100)$	$dmm_i(500)$	$dmm_i(1000)$
τ_{12}	1	2	3	4	4
τ_{13}	1	2	4	4	4
τ_{26}	1	3	4	4	4

Table 5. DMMs for OBSW tasks when FPP scheduling is considered.

8.2 Synthetic test cases

In this section, we present a set of synthetic test cases that we have developed to test more extensively our approach on a variety of systems. We aim to study and illustrate the impact of EDF scheduling on DMMs by comparing the impact of different factors on the computation of $dmm_i(k)$

when EDF and RM are considered respectively. Particularly, we study the utilization U, the share of sporadic overload U_O/U where $U_O = \sum_{s \in O} U_s$, and the system size.

Synthetic test case generation. Our synthetic test cases consist of a set of tasks with a worst-case execution time, a period (for periodic tasks), a minimum distance function (for sporadic tasks), and a relative deadline. Every task set $\mathcal{Z}$ consists of typical tasks that are chosen to be periodic in this experiments, and overload tasks, i.e. $\mathcal{Z} = \mathcal{T} \cup \mathcal{O}$. The following steps summarize how we generated them:

- We first choose the number of tasks $n = \#\{\mathcal{Z}\}$ and the number of overload tasks $n_s = \#\{\mathcal{O}\}$. We then decide on the system utilization to be shared among the tasks.
- UUnifast [5] is applied to assign a share of the system utilization to each task (sporadic or periodic, typical or overload).
- For typical tasks, we assign periods randomly chosen in a predefined set of harmonic values. Then, the worst-case execution time of typical tasks is computed as follows: $C_i = U_i * p_i$, where p_i denotes the period of the typical task τ_i. Note that $\delta_i^-(n) = (n-1)\ .\ p_i$.
- Generating minimum distance functions for sporadic tasks is not straightforward, and there is no standard approach for this. In particular, they have to be super-additive, i.e., $\delta^-(a+b) \geqslant \delta^-(a) + \delta^-(b)$ for all a, b [23]. To achieve this, we depended on the definition of U_s of a sporadic task, which is defined as follows (see Definition 3.3):

$$U_s = \lim_{n \to \infty} \frac{(n-1) * C_s}{\delta_s^-(n)}$$

  We first randomly assign the worst-case execution time for each sporadic task $\tau_s \in \mathcal{O}$ such that $C_s \in [\min_{i \in \mathcal{T}}\{C_i\}, \max_{i \in \mathcal{T}}\{C_i\}]$. The justification of this choice is not to bias the activation model of τ_s toward having a high density of jobs by selecting short execution times or low density by selecting long execution times. Then, we compute $\delta_s^-(N) = (N-1) * C_s/U_s$, with a sufficiently large N (for all our experiments $\delta_s^-(100)$ is much larger than the longest busy-window). We generated then a trace of N jobs such that the first one is at 0 and the last one is at $\delta_s^-(N)$ and $N-2$ jobs in between. We use pyCPA [15] to extract the minimum distance function from the generated trace (*model.TraceEventModel(trace)*).

To guarantee a wide variety of system models, we chose the various parameter values as follows:

- Number of tasks $n \in [3, 45]$ and number of sporadic overload tasks $n_s \in [1, 20]$. On the one hand, analyzing a system with $n < 3$ makes no sense because it is trivial. On the other hand, scheduling 45 tasks on one resource is realistic and acceptable.
- Total utilization is $U \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Resources in the generated test cases range from average to maximum allowed utilization.
- The transient overload U_O has a share of $U_O/U \in \{0.001, 0.01, 0.05, 0.1, 0.15, 0.2\}$. The generated test cases range from being slightly overloaded to highly overloaded.
- Each typical task is assigned a relative deadline $D_i \in \{0.6, 0.8, 1, 1.2, 1.4\} \times p_i$. Thus, the deadline is arbitrary.
- Each overload task is assigned a relative deadline $D_s = C_s$ to get higher priorities under EDF scheduling.

Note that we chose to have a range of values rather than just intervals to facilitate the study of parameters impact.

Results. We performed the experiment over 5000 task sets generated as explained above. For each task set, we applied first the EDF scheduling policy, and we computed $dmm_i(k)$ for $k = 10, 100, 500, 1000$ where $\tau_i \in \mathcal{T}$. Then we assigned the highest priorities to the sporadic overload

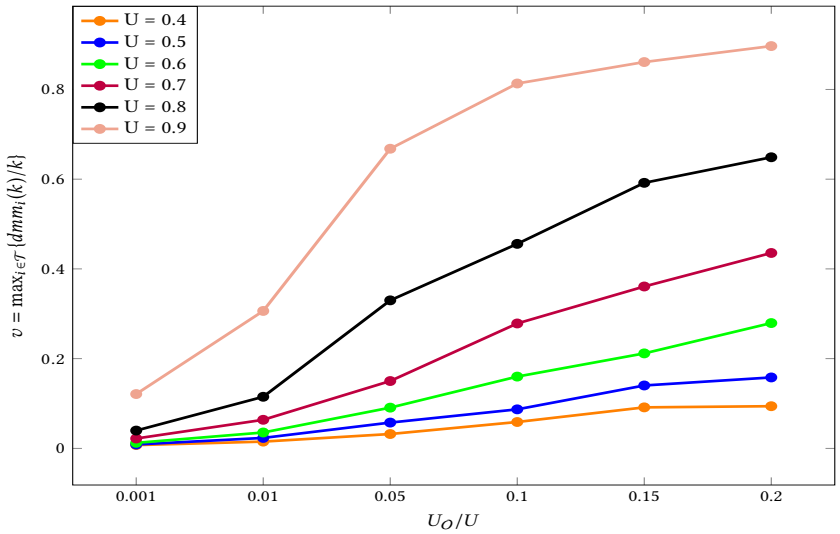

Fig. 11. The impact of U and U_O/U on the computed DMM under EDF scheduling.

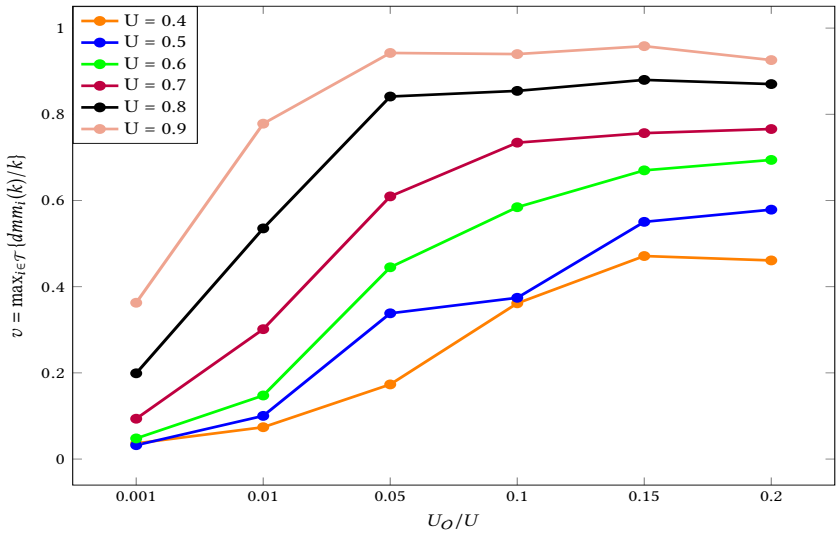

Fig. 12. The impact of U and U_O/U on the computed DMM under RM scheduling.

tasks to interfere with all tasks in $\mathcal{T}$, and we assigned fixed priorities to tasks in $\mathcal{T}$ according to RM. We aim to show a comparison between fixed priority scheduling and EDF. As the relative deadline can be larger than the period (D > T) neither RM, which has been proven by Liu and Layland [29] to be optimal when $D = T$, nor DM, which has been shown by Leung and Whitehead [26] to be optimal when $D \leqslant T$, is optimal. However, we used RM as a systematic approach to assigning the priorities instead of random assignment. Studying the impact of priority assignments, e.g. DM, RM, EDF, on the computed DMM is an interesting topic. However, this is out of the scope of this paper.

TWCA does not apply to task sets in which $\mathcal{T}$ is not schedulable. The 5000 generated task sets have $\mathcal{T}$ schedulable under EDF. However, there are 398 task sets in which $\mathcal{T}$ is unschedulable under RM.

Now we have 4602 task sets to continue the experiment. we normalize $dmm_i(k)$ to 1 by reporting $v = \max_{i \in \mathcal{T}}\{dmm_i(k)/k\}$. That facilitates the comparison between systems with different number of tasks and between $dmm_i(k)$ for different k.

Figure 11 shows the average of the reported value v under EDF. Each curve represents a certain utilization. Each point of the curves is the average of v of task sets that have the same U and the same U_O/U. Curves grow with increasing U_O/U and they are shifted up with increasing U.

We can draw the same conclusion from Figure 12 where RM is considered. By comparing both figures, we conclude that EDF can help tasks to miss fewer deadlines and tolerate larger overload. However, we showed in the last experiment (OBSW) that EDF distributes the deadline-misses causing all tasks in $\mathcal{Z}$ to miss their deadline. This experiment supports this conclusion where in 3050 task sets out of 4602 all tasks in $\mathcal{Z}$ miss their deadline when EDF is considered. All tasks in $\mathcal{Z}$ miss their deadline in 1517 task sets when RM is considered. Remember that every sporadic overload task has $D_s = C_s$ in both EDF and RM. The results are therefore pessimistic. In RM, the highest priority task misses its deadline when $\delta_s^-(2) < C_s$, such a situation can be considered as a *burst*.

We reported in this experiment the average running time to compute $dmm_i(500)$ over 4602 task sets for RM and EDF. When RM is considered the average running time was $0.623s$, while it was only $0.03379s$ for EDF. To explain this result, let us recall the set of unschedulable combinations $\tilde{C}$. The size of this set determines the number of variables (columns) of our ILP, or the number of steps in the column generation algorithm, which is used in the LP relaxation. A task set $\mathcal{Z}$ does not have the same $\tilde{C}$ under EDF and RM. Let $\tilde{C}^{EDF}$ denotes the set of unschedulable combinations under EDF and $\tilde{C}^{RM}$ under RM; it is always true that $\tilde{C}^{EDF} \subseteq \tilde{C}^{RM}$ because every schedulable combination $\bar{c}$ under RM is schedulable under EDF but not vice versa. The experiment over 4602 shows that computing $dmm_i(k)$ under RM required at most 361 steps (columns to be generated), while under EDF it required at most 17 steps.

At this point, we can propose the following conclusions:

- TWCA reports pessimistic bounds. The sources of pessimism are illustrated in Section 7.
- TWCA scales well w.r.t. k, n and n_s. The experiment covers $k = 10, 100, 500, 1000$, $n \in [3, 45]$, and $n_s \in [1, 20]$ with a maximum running time equals to $1.3s$ for EDF and $32.5s$ for RM.
- TWCA implies better to a relatively low transient sporadic overload. Figures 11 and 12 illustrate this conclusion.
- EDF can help tasks to miss fewer deadlines and tolerate larger overload comparing to RM.
- EDF is proper for systems that comprise only WHRT tasks. RM is proper for systems in which hard and weakly-hard real-time tasks are defined.

9 CONCLUSION

Many practical real-time systems are weakly-hard, which can de facto sustain a bounded number of deadline-misses. That makes computing WHRT guarantees to be a requisite as the worst-case guarantees are not expressive for these systems. In this work, we addressed the problem of computing WHRT guarantees in the form of a DMM using TWCA for independent tasks where EDF scheduling policy is considered.

Through the pages of this paper, we first recalled state-of-the-art worst-case response time analysis of EDF which is the foundation of TWCA. Later, we presented state-of-the-art TWCA to

compute DMMs. We showed then how to adopt TWCA to compute DMMs for independent tasks under EDF scheduling policy in temporarily overloaded systems.

This work was motivated by and validated on a realistic case study inspired by industrial practice (satellite on-board software) and on a set of synthetic test cases. The experiment was performed over 5000 task sets to compare DMMs computed under EDF and RM scheduling policies. The goal was to study the impact of EDF scheduling on DMMs extensively. The results showed that EDF can help tasks to miss fewer deadlines and tolerate larger overload comparing to RM. They also showed that EDF distributes the deadline-misses among tasks. Therefore, EDF will likely lead to failure to the hard real-time tasks.

This works can be used for considering distributed WHRT systems, where heterogeneous scheduling policies can be used, in order to provide end-to-end weakly-hard real-time guarantees such as end-to-end DMMs.

REFERENCES

[1] Avizienis A, Laprie JC, Randell B, Landwehr C (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing 1(1):11–33, DOI 10.1109/TDSC.2004.2

[2] Axer P, Sebastian M, Ernst R (2011) Reliability analysis for mpsocs with mixed-critical, hard real-time constraints. In: Proc. Intl. Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Taiwan

[3] Bernat G, Cayssials R (2001) Guaranteed on-line weakly-hard real-time systems. In: Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (Cat. No.01PR1420), pp 25–35, DOI 10.1109/REAL.2001.990593

[4] Bernat G, Burns A, Llamosí A (2001) Weakly hard real-time systems. IEEE Trans Computers 50(4):308–321

[5] Bini E, Buttazzo G (2005) Measuring the performance of schedulability tests. Real-Time Systems 30(1-2):129–154, DOI 10.1007/s11241-005-0507-9, URL http://dx.doi.org/10.1007/s11241-005-0507-9

[6] Bini E, Buttazzo G (2009) The space of edf deadlines: the exact region andÂăaÂăconvex approximation. Real-Time Systems 41(1):27–51, DOI 10.1007/s11241-008-9060-7, URL https://doi.org/10.1007/s11241-008-9060-7

[7] v d Brüggen G, Chen KH, Huang WH, Chen JJ (2016) Systems with dynamic real-time guarantees in uncertain and faulty execution environments. In: 2016 IEEE Real-Time Systems Symposium (RTSS), pp 303–314, DOI 10.1109/RTSS.2016.037

[8] Buttazzo GC (2011) Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, 3rd edn. Springer Publishing Company, Incorporated

[9] Carnevali L, Melani A, Santinelli L, Lipari G (2014) Probabilistic deadline miss analysis of real-time systems using regenerative transient analysis. In: Proceedings of the 22Nd International Conference on Real-Time Networks and Systems, ACM, New York, NY, USA, RTNS '14, pp 299:299–299:308, DOI 10.1145/2659787.2659823, URL http://doi.acm.org/10.1145/2659787.2659823

[10] Cervin A (2001) Analyzing effects of missed deadlines in control systems. In: ARTES Graduate Student Conference

[11] Chakraborty S, Künzli S, Thiele L (2003) A general framework for analysing system properties in platform-based embedded system designs. In: Proceedings of DATE'03, IEEE Computer Society, pp 190–195

[12] Chen KH, Chen JJ (2017) Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In: 2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES), pp 1–8, DOI 10.1109/SIES.2017.7993392

[13] Dertouzos ML (1974) Control robotics: The procedural control of physical processes. In: IFIP Congress, pp 807–813

[14] Diaz JL, Garcia DF, Kim K, Lee CG, Bello LL, Lopez JM, Min SL, Mirabella O (2002) Stochastic analysis of periodic real-time systems. In: 23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002., pp 289–300, DOI 10.1109/REAL.2002.1181583

[15] Diemer J, Axer P, Ernst R (2012) Compositional performance analysis in python with pycpa. In: 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)

[16] Geelen W, Antunes D, Voeten JPM, Schiffelers RRH, Heemels WPMH (2016) The impact of deadline misses on the control performance of high-end motion control systems. IEEE Transactions on Industrial Electronics 63(2):1218–1229, DOI 10.1109/TIE.2015.2504339

[17] George L, Rivierre N, Spuri M (1996) Pre-emptive and nonpre-emptive real-time uni-processor scheduling. Tech. rep., INRIA, France

[18] Guan N, Yi W (2014) General and efficient response time analysis for edf scheduling. In: 2014 Design, Automation Test in Europe Conference Exhibition (DATE), pp 1–6, DOI 10.7873/DATE.2014.268

[19] Hamdaoui M, Ramanathan P (1995) A dynamic priority assignement technique for streams with (m, k)-firm deadlines. IEEE Trans Computers 44(12):1443–1451

[20] Hammadeh ZAH, Quinton S, Ernst R (2014) Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In: Proceedings of the 14th International Conference on Embedded Software, ACM, EMSOFT '14, pp 10:1–10:10, DOI 10.1145/2656045.2656059
[21] Hammadeh ZAH, Ernst R, Quinton S, Henia R, Rioux L (2017) Bounding deadline misses in weakly-hard real-time systems with task dependencies. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2017, pp 584–589, DOI 10.23919/DATE.2017.7927054
[22] Hammadeh ZAH, Quinton S, Panunzio M, Henia R, Rioux L, Ernst R (2017) Budgeting under-specified tasks for weakly-hard real-time systems. In: The 29th Euromicro Conference on Real-Time Systems (ECRTS17), Dubrovnik, Croatia, URL http://drops.dagstuhl.de/opus/frontdoor.php?source_opus=7163
[23] Henia R, Hamann A, Jersak M, Racu R, Richter K, Ernst R (2005) System level performance analysis — the SymTA/S approach. In: IEEE Proceedings Computers and Digital Techniques
[24] Horn WA (1974) Some simple scheduling algorithms. Naval Research Logistics Quarterly 21(1):177–185, DOI 10.1002/nav.3800210113, URL http://dx.doi.org/10.1002/nav.3800210113
[25] Kumar P, Thiele L (2012) Quantifying the effect of rare timing events with settling-time and overshoot. In: Proceedings of RTSS'33, pp 149–160
[26] Leung JYT, Whitehead J (1982) On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance Evaluation 2:237–250
[27] Li J, Song Y, Simonot-Lion F (2004) Schedulability analysis for systems under (m,k)-firm constraints. In: IEEE International Workshop on Factory Communication Systems, 2004. Proceedings., pp 23–30, DOI 10.1109/WFCS.2004.1377670
[28] Lima G, Burns A (2005) Scheduling fixed-priority hard real-time tasks in the presence of faults. In: Proceedings of the Second Latin-American Conference on Dependable Computing, LADC'05, pp 154–173, DOI 10.1007/11572329_14
[29] Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM 20(1):46–61, DOI 10.1145/321738.321743, URL http://doi.acm.org/10.1145/321738.321743
[30] Locke CD (1986) Best-effort decision-making for real-time scheduling. PhD thesis, Carnegie Mellon University
[31] López JM, Díaz JL, Entrialgo J, García DF (2008) Stochastic analysis of real-time systems under preemptive priority-driven scheduling. Real-Time Systems 40(2):180–207
[32] Magalhães AP, Rela MZ, Silva JG (1993) Deadlines in real-time systems. Tech. rep., Universidade do Porto, Portugal, URL https://web.fe.up.pt/~apmag/Suportehome/Ficheiros/drts.pdf
[33] Magazine MJ, Chern MS (1984) A note on approximation schemes for multidimensional knapsack problems. Mathematics of Operations Research 9(2):244–247, DOI 10.1287/moor.9.2.244
[34] Natale MD (2017) Beyond the m-k model: restoring performance considerations in the time abstraction. ESWEEK - Tutorial Slides
[35] Pazzaglia P, Di Natale M, Buttazzo G, Secchiari M (2018) A framework for the co-simulation of engine controls and task scheduling. In: Cerone A, Roveri M (eds) Software Engineering and Formal Methods, Springer International Publishing, Cham, pp 438–452
[36] Quan G, Hu X (2000) Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In: Proceedings 21st IEEE Real-Time Systems Symposium, pp 79–88, DOI 10.1109/REAL.2000.895998
[37] Quinton S, Hanke M, Ernst R (2012) Formal analysis of sporadic overload in real-time systems. In: Proceedings of DATE'12, IEEE, pp 515–520
[38] Ramanathan P (1999) Overload management in real-time control applications using (m, k)-firm guarantee. IEEE Transactions on Parallel and Distributed Systems 10(6):549–559, DOI 10.1109/71.774906
[39] Richter K (2005) Compositional scheduling analysis using standard event models. PhD thesis, TU Braunschweig
[40] Santinelli L, Cucu-Grosjean L (2015) A probabilistic calculus for probabilistic real-time systems. ACM Trans Embed Comput Syst 14(3):52:1–52:30, DOI 10.1145/2717113, URL http://doi.acm.org/10.1145/2717113
[41] Shin K, Krishna C, Lee YH (1985) A unified method for evaluating real-time computer controllers and its application. IEEE Transactions on Automatic Control 30(4):357–366, DOI 10.1109/TAC.1985.1103952
[42] Spuri M (1996) Analysis of deadline schedule real-time systems. Tech. rep., INRIA, France
[43] Sun Y, Lipari G (2015) Response time analysis with limited carry-in for global earliest deadline first scheduling. In: 2015 IEEE Real-Time Systems Symposium, pp 130–140, DOI 10.1109/RTSS.2015.20
[44] Sun Y, Natale MD (2017) Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. International Conference on Embedded Software (EMSOFT), ACM Transactions on Embedded Computing Systems ESWEEK Special Issue
[45] Xu W, Hammadeh ZAH, Kröller A, Ernst R, Quinton S (2015) Improved deadline miss models for real-time systems using typical worst-case analysis. In: 2015 27th Euromicro Conference on Real-Time Systems, pp 247–256, DOI 10.1109/ECRTS.2015.29
[46] Zhang F, Burns A (2009) Schedulability analysis for real-time systems with edf scheduling. IEEE Trans Comput 58(9):1250–1258, DOI 10.1109/TC.2009.58, URL http://dx.doi.org/10.1109/TC.2009.58